Cooperative CEP-based RFID framework: a notification approach for sharing complex business events among organizations

Leonardo A. Amaral[1], Fabiano P. Hessel[1], Jerônimo C. Corrêa[1]
Embedded Systems Group (GSE)
Catholic University of Rio Grande do Sul (PUCRS), Porto Alegre - Brazil
[1]{leonardo.amaral, fabiano.hessel, jeronimo.correa}@pucrs.br

Abstract – Through an organizational perspective, an RFID system must provide services for intelligent data management and business events integration among organizations. The widely adopted EPC Network fails to adequately support application services for event stream capture and lacks event communication methods to provide indirect exchange of events in distributed activities of complex event detection. This work presents an RFID software framework whose purpose is to raise the inter-organizational integration of an RFID middleware through cooperative complex event processing (CEP) mechanisms based on event notification services. Experimental results show that the framework improves activities of design and development of RFID applications and adds new features for integrating, managing and sharing RFID event data.

Keywords - *RFID Middleware, Complex Event Processing, CEP-based RFID Framework, Cooperation Services, CEP Network.*

I. INTRODUCTION

Over the past ten years, the ability of RFID technology to identify uniquely individual objects without line of sight has enabled many new kinds of applications.

Industry has adopted a standard RFID middleware architecture based on the EPCglobal Application Level Events (ALE) standard. However, this commonly used specification framework is inefficient in the semantic processing of event data because it does not provide effective support for complex event processing (CEP).

This deficiency requires application developers and systems integrators to create data processing solutions in their on applications when they need to derive significant events from event data delivered by a conventional RFID middleware. In this sense, the use of CEP integrated with some RFID processing technology, helps discover complex events by analyzing the occurrence of other events on the same business environment. CEP has the ability to process data streams from multiples read events aiming at the transformation of basic events into meaningful events that derive effective knowledge for decision-making.

Apart from the limitations of conventional mechanisms for managing RFID data, existing specifications for RFID software technology (EPC Network according to EPCglobal) have some challenges in the services proposed to integration and sharing of events among different business organizations (ONS, EPCIS and DS) [1]. These services do not support methods of indirect communication of events, which difficult the detection of distributed complex events.

This happens because EPCIS information service only supports direct addressing of resources and demand constant adjustment of the communication links between request application and information sources, which degrades the system due the large amount of communication required.

As a proposal of improvements to above problems, this paper presents an RFID software framework whose purpose is to raise inter-organizational integration capability of an RFID middleware system based on CEP mechanisms that cooperate with each other through event notification services. The CEP technology is integrated with an RFID middleware that implements the specifications of ALE according to EPCglobal. The framework is based on the concept of CEP network for processing and notification of distributed complex events among different organizations. Each processing node has its own CEP mechanism and the cooperation between them provides a system of global business rules.

Experimental results show that our framework improves activities of design and development of RFID applications and adds new features for integrating, managing and sharing RFID event data.

The remainder of this paper is organized as follows. In section 2 we present related work. Section 3 presents the architecture and information flow overview of our framework. Section 4 and 5 presents experimental results and an analysis. And section 6 presents the conclusions and future work.

II. RELATED WORK

A. RFID Data Management Systems

One of the challenges of RFID applications is the data management capability. This subject is important because RFID event data have specific characteristics and require special treatment [2]. Large-volume of data streams, inaccuracy, temporal and spatial data and implicit semantic, are typical examples of RFID event data.

Fortunately, the scientific community has been developing solutions to issues around management data in RFID systems [3] [4] [5] [6]. Technologies such as Siemens RFID Middleware, MDI-Smurf Middleware, EPC-Bitmap, UIUC's RFID Middleware, REFILL, LIT Middleware, RF^2ID and MARM are some of the major existing systems. Most of these systems support the processing of large-volume and inaccurate data streams. However, only Siemens RFID middleware addresses issues around modeling

temporal and spatial data, implicit references and limited active lifespan [3].

Many systems are still focused on improving the reliability of event data reading and do not address problems related to semantic transformation of RFID event data.

B. RFID Data Integration Systems

Another challenge of RFID technology is the integration of large-scale RFID applications, which has challenging requirements such as scalability, heterogeneity, manageability and flexibility.

Major software development companies have been working on developing systems for integration of RFID applications, such as: SAP AII [7] WinRFID [8], Sun Java System RFID Software [9], and Fosstrak [10]. Many of these systems have basic functions of data management and do not support the processing of RFID complex event (CEP).

Scalability, heterogeneity and openness are examples of the main requirements supported by most of theses systems. However, they have very limited resources available to issues around flexibility and manageability. Several architectures provide distributed functionalities, and few integrate RFID event data with business processes through rule-based engines. Moreover, only Fosstrak presents solutions to fill the lack of scalability and flexibility of the capture application specified by EPCglobal.

C. CEP-based RFID Systems

The CEP technology, which was designed specifically to address issues around processing real-time events in distributed systems, has been studied extensively in the past [11]. Some studies are focused on detecting complex events for rule-based business activity management in ubiquitous computing environment [12]. However, these studies are generic in nature, and do not address characteristics that are unique to RFID technology.

With the rapid development of RFID technology, more studies have started to explore the use of CEP in RFID data management activities. The traditional ECA model [13], based on active database systems, could not be directly applied to RFID systems because RFID data is temporal and in large volume.

Other works provided system solutions to integrate RFID and non-RFID data to detect complex events [14] [20]. Moreover, different aspects were also addressed, such as: techniques for semantic data processing of RFID events [15] [16], CEP-based RFID middleware for large-scale and high-performance systems [17], and frameworks for management RFID events [18].

III. CEP-BASED RFID SOFTWARE FRAMEWORK

A. Architecture and Features Overview

Figure 1 presents an overview of the RFID framework proposed in this paper. The software elements were created according to the EPCglobal standards and enable a seamless integration with EPC Network compliant systems. The key elements of the framework are: RFID Middleware, EPCIS Information Service and CEP Mechanism.

The CEP mechanism contains a rule engine, which is able to process streams of complex events. The rule engine is integrated with Drools, a system manager of business rules. Rules created in the CEP module are described according to the language defined by Drools.

RFID events (*ECReports*) are captured through a capture service (ALE Capture Web Service) and transformed into business events (EPCIS events) through the rule engine. After the event contextualization, these events are stored in the information service (EPCIS).

The event capture service (CEP Capture Web Service) allows CEP to receive events from other capture applications (other sources of data or other CEP modules) to be processed, stored, or to generate new business events. Captured events are managed through an event handler module that sends events to the rule engine. The event handler module also communicates with other instances of middleware (ALE) and information services (EPCIS) through web services that implement the ALE and EPCIS Interfaces (Capture and Query interface).

The ALE interface is responsible for the integration between CEP mechanism and RFID middleware (ALE). It is able to define and execute reading cycles of events as well as to manage logical readers. ALE interface allows inclusion, modification and removal of logical readers.

The EPCIS interface is responsible for communication with query manager and query interface of the EPCIS repository. The query interface allows CEP engine to use the records of the repository for detection of complex events as well as to conduct the requests of the application for relevant information.

The query manager uses the EPCIS interface as a communication channel with the EPCIS repository. It organizes application requests in order to reduce problems of communication overhead caused by activities of query events in several information services (EPCIS).

CEP can hold one or more ALE and EPCIS interfaces, which is managed by the administrative interface. This interface is responsible for managing the processes of CEP (rules and business processes). This management is regulated through organizational profiles, which define the business logic described in the rules. Each profile contains a set of interfaces (ALE and EPCIS) and rules belong to one or more business processes. Through the profiles, the architecture offers a way of event integration, both intra and inter-organizational. Moreover, it organizes the data hierarchically decomposing rules according to the profile it belongs.

CEP has a set of event notification services that is divided into application and CEP network services. These services notify events to business partners involved (applications or CEP network nodes), enable search for information of events (EPCIS events), among other features. The notification services permits that applications receive only events of interest.

B. Notification Service for Collaborative Event Processing

Most distributed applications today receive events, process them and in turn create new events, which are sent to other processes or applications. Basic events, potentially occurred at different sites, are correlated in order to detect complex event patterns formed by basic events. The possibility to federate and cooperate events offers the opportunity to pool resources together and share data for common benefits.

Integrating RFID data from various sources in a value chain is an important requirement for leveraging the

potential of RFID technology. To facilitate this crucial integration, EPCglobal are currently developing a range of service specifications (ONS, DS and EPCIS). Yet, the services specified so far do not enable event-based interaction schemes and do not leverage the detection of distributed complex events.

In our work, the approach used to promote the event-based communication scheme is founded on event notification services (Figure 2). Applications can define the decomposition of business rules between CEPs of different organizations enabling the detection of distributed complex events. Whenever a business rule is defined and split between internal nodes of the organization, this rule is not registered on other CEPs and the processing is local and hierarchical among the elements involved. However, when a rule requires the detection of inter-organizational events, the rule components are specified in each local CEP involved. It reduces the communication overhead caused by activities of querying events in the processing of distributed events.

C. *Drools-based CEP Engine*

We selected the open source software Drools 5.0 [19] as our CEP engine. Drools supports complex event processing of multiple events from an event cloud for event detection, correlation, and abstraction.

It has several advantages. First, it supports asynchronous multi-thread streams in which events may arrive at any time and from multiple sources. Second, since temporal reasoning is an essential part of CEP, we examined the capability of Drools in support of temporal relationships. It has a complete set of temporal operators to allow modeling and reasoning over temporal relationships between events. Third, it allows complete and flexible reasoning over the absence of events (i.e., negation). Lastly, sliding windows supports aggregation of events over temporal or length-based windows.

Drools is based on a forward chaining inference engine that uses an enhanced implementation of RETE algorithm. RETE algorithm is used in expert systems for production-based logical reasoning, matching a set of facts against a set of inference rules. By default it does not support temporal operators.

In our work, we are not describing the details of the RETE algorithm [21] since we focus on the extension of RETE with temporal reasoning supported by Drools engine events constructors. To support temporal constraints in rules, events can be modeled as facts with timestamps and they are inserted into the engine working memory at runtime. In addition, events need to be discarded when they are no longer of interest or cannot contribute to complex events.

D. *CEP Preliminaries*

In our work, complex event processing (CEP) has been introduced to process and correlate RFID complex event data. This technique aims at processing multiple streams of data continuously and identifying meaningful events in real-time. In this subsection, we formalize the definition of events, event constructors, and CEP rules. The relationship of CEP concepts is illustrated in figure 3.

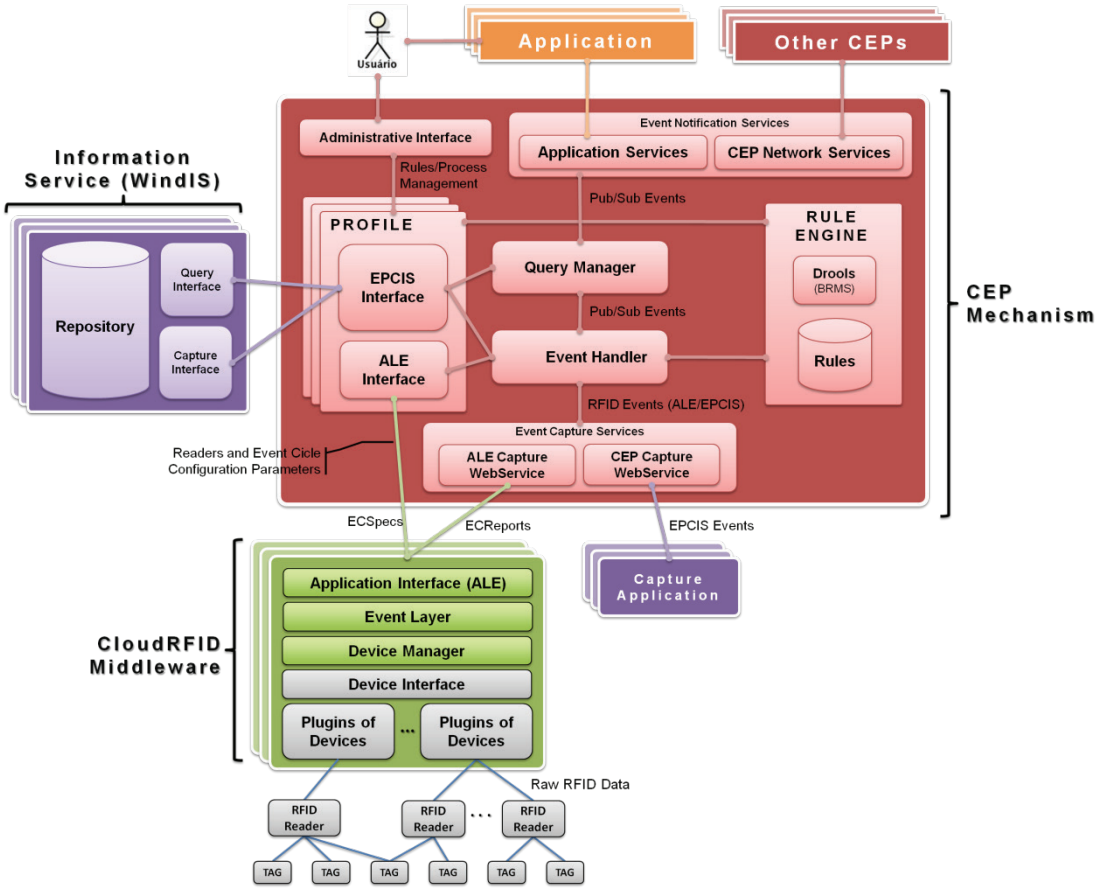

Figure 1: CEP-based RFID framework architecture overview.

1) Event and Event Constructors

An event can be defined as a record of an activity in a system for computational purpose. In general, events can be categorized into basic event and complex event. We use upper case (E) and lower case (e) to represent event type and event instance, respectively.

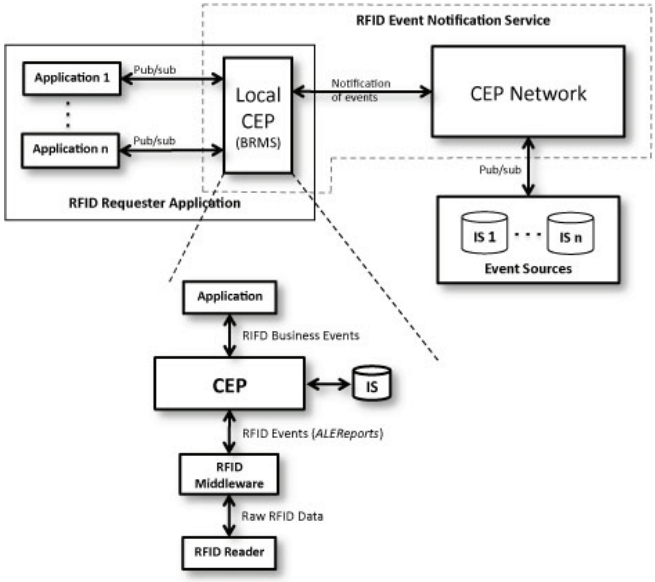

Figure 2: Event Notification Service overview.

A basic event can be defined as $E= E\ (id, a, t)$, where *id* is the unique ID of an event, $a= \{a_1, a_2, \ldots a_n\}\ n > 0$ is a set of event attributes and *t* is the event *occurrence time*. A basic event is atomic, indivisible, and occurs at a point in time.

An RFID event is an instance of a basic event that can be denoted as $E= e\ (id, r, t)$, where *id* is the tag EPC, *r* is the reader ID that captures this tag, and *t* represents the observation *timestamp*.

A complex event can be defined as $E= E\ (id, a, c, t_b, t_e)$, $t_e >= t_b$, where *id* is the unique ID of an event, $a= \{a_1, a_2, \ldots a_n\}\ n > 0$ is a set of event attributes, $c= \{e_1, e_2, \ldots e_n\}, n>0$ is the vector that contains the basic events and complex events that cause this event happen, t_b and t_e are the starting and ending times of this complex event. It can happen over a period of time (i.e., from t_b e t_e).

Complex events are aggregations of basic events or complex events using a specific set of event constructors such as disjunction, conjunction, and sequence. It signifies or refers a set of other events to indicate a situation described in the application scenario. Complex events contain more semantic meaning and are more useful for decision making in business applications.

Event constructors or event operators are used to express the relationships among events and correlate events to form complex events. In [3], the author's give a comprehensive set of event constructors and classify them into temporal and non-temporal constructors.

2) CEP Rules

Based on the formalization of events and event constructors, CEP rules are defined to specify domain syntax and semantics. A rule is a predefined inference logic or pattern for detecting complex events.

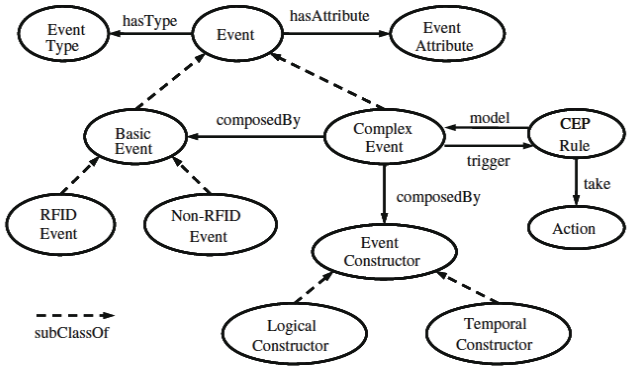

Figura 3: CEP related concepts.

Several studies have described different syntax for CEP rules [2] [3] [12]. In our work, we create and integrate business rules using the declarative language of events provided by Drools engine. Drools uses ECA (event–condition–action) rule expression language to describe event patterns since it is easier to use and more understandable.

The generic syntax of ECA can be expressed as follows:

Rule rule_id, rule_name
ON *event*
IF *condition*
THEN *action1... action* n
end

Where *rule_id* and *rule_name* are unique for each rule, suggesting the *id* and *name* for a rule; event specifies the event of interest; condition is a *boolean* combination of user-defined functions; *action* defines a user-defined procedure (e.g., to send out alarms) or an update in the database (e.g., update of a product status). With CEP rules, we can provide sufficient support for processing RFID and other sensor data, such semantic data filtering and real-time monitoring.

E. Modeling Events in an RFID-enabled Smart Store

An RFID-enabled smart store application can generate a variety of data streams and need to be processed in a timely manner. The system consistently generates data about the location and time of tagged items, which is in low-level semantics and not directly useful.

A rule example can be the creation of EPCIS events based on *ALEReports* received from RFID middleware. The first rule creates a generic EPCIS event that is used for the second rule to *ALEReports* contextualization.

```
rule "Create EPCIS event from ECReports" dialect "java"
    when
      $reports : ECReports($specName : specName)
      $epcs : LinkedList(size > 0) from collect (
      EPC() from getReadInformation($reports))
      $readPoint : ReadPoint(name == $specName)
    then
      EPCISEventType event = createEpcisEvent($readPoint,
      $epcs);
      insert(event);
      retract($reports);
end
```

The second rule receives generic events from an RFID middleware and contextualizes them into an EPCIS event. This business event has an invoice used to check the objects

related with the current event business step. So, this action is represented by the third rule.

```
rule "Cast to transaction EPCIS event" dialect "java"
    when
      $event : EPCISEventType()
      eval($event instanceof TransactionEventType)
    then
      TransactionEventTypetransactionEvent=
      (TransactionEventType) $event;
      retract($event);
      insert(transactionEvent);
end

rule "Check items against invoice" dialect "java"
    when
      $event : TransactionEventType($parentID : parentID)
      $invoice : EletronicInvoice(id == $parentID)
                 not(eval(check($event, $invoice)))
    then
      TransactionEventTypeepcisEvent=
      createTransactionFailed($event, $invoice);
      captureEvent("Retailer", epcisEvent);
      retract($event);
      retract($invoice);
      insert(epcisEvent);
end
```

The fourth rule is an example of using temporal operators. This complex rule establishes a fail transaction scenario where an action event is created if a transaction event does not happen after a fixed time (i.e. one hour and thirty minutes).

```
rule "Failed delivery"
    when
      $invoice : EletronicInvoice(id == $parentID)
      not($event:TransactionEventType($parentID:
      parentID,
      this after[0s,1h30m] $invoice))
    then
      TransactionEventTypeepcisEvent=
      createTransactionFailed($event, $invoice);
end
```

IV. Experimental Results

In this section, our CEP-based RFID Software Framework is evaluated. The chosen scenario for this validation is a smart store application.

A. Performance Evaluation

The tests were applied in an application prototype for an RFID smart store. We used our software artifacts to create this application. The prototype was simulated in laboratory using sets of synthetic data tests (ALE Reports) generated from the simulation of reading events of RFID tags.

1) Test Objectives

The tests have two main objectives. The first is to identify points of performance vulnerability in the proposed CEP mechanism. The second is to evaluate the artifacts and mechanisms that comprise the framework applied to activities of design and development of RFID applications.

The evaluations were made through tests of the prototype and comparisons with reference architectures, as the project Fosstrak [10], for example.

2) Test Methodology

The test methodology is based on two test scenarios within the same prototype application. The first scenario is stimulated by a smaller set of simple business rules (not involving temporal and relational operators). The second test scenario is driven by a larger set of complex business rules involving inter-relationship of RFID event data (local or remote EPCIS event data) and reference data (information of product invoices for example).

To avoid communication overheads in the steps of test data generation (network delays), this step was done directly using the framework API through the submission of files containing simulated events of RFID data.

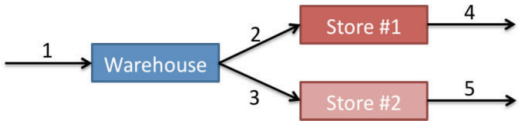

Figure 4: Application test prototype.

The application prototype consists of one RFID warehouse and two stores (Figure 4). The arrow 1 represents products being sent to storage in the warehouse, while arrows 2 and 3 represent requests of products for stock replenishment in the stores 1 and 2. Arrows 4 and 5 represent products leaving the stores.

The model of test data generation was defined following some rules, which determine the behavior of the prototype dictating the occurrence of events. Examples of this model are:

- From the arrival of goods in the warehouse these products pass through several stages into the business process involved (Figure 5). Upon inspection, 10% of products have some problems and are removed from the model.
- After the arrival of goods in the stores (Figure 6), these products are stored in an internal stock and according to the quantity of products displayed for sale, they are being transferred to the shelves. The products may leave the shelves together or in units. 80% of these products go to the dressing room, 17% are directly purchased and 3% are attempted robbery.
- When one or more products enter the dressing room, the application queries the back-end information system to suggest other products that have already been sold with the products detected in the dressing room. When a purchase is made, the POS (Point of Sale) disables the RFID tags and the products are removed from the simulation model.

Figures 5 and 6 show internal processes of the warehouse and the stores. In the figure 5, each Read Point (RP) is an RFID portal (Door Way) where the products go through. Each room in the warehouse (Business Location) is a business process step. Steps 0 to 5 show a basic flow of a product in the store since its arrival, inspection, storage and shipping.

Figure 6 contains a basic outline of the structure of the stores. Points RP:R, RP:S, and RP:E represent RFID portals. While the points RP:P represents shelves, RP:D dressing rooms, and RP:C points of sale (POS). Steps 5 to 9 represent

a basic flow of a product since its arrival in the store until it is sold.

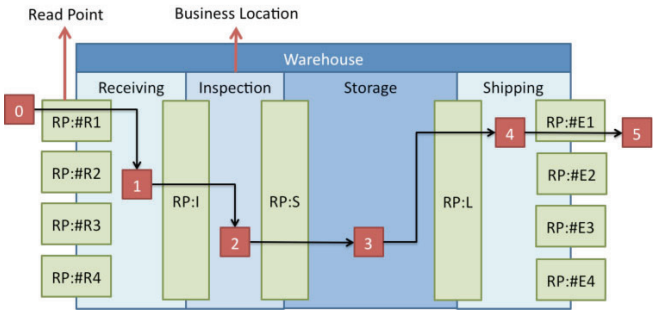

Figure 5: Process steps of the Warehouse.

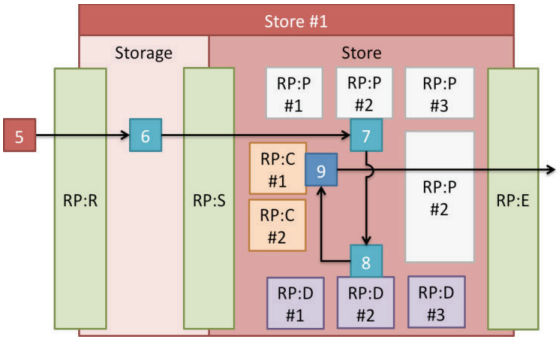

Figure 6: Process steps of the Stores.

The first test scenario is related to business processes of the warehouse (Figure 5) whose rules involved are related to event queries for unsuccessful delivery of products. The second scenario is related to all process of the prototype (one warehouse and two stores, according to figure 4).

In the second scenario, the rules contain expressions to detect complex events and events containing correlations of past events, such as historical data of products that were detected in the dressing room. The expressions used for the detection of complex events include at least one logical operator (AND, OR and NOT) and a temporal operator, and may also be correlated with other kind of events.

The events generated from the created rules are stored in the information service (EPCIS) and can be used to track and trace products or to correlate rules with some kind of historical events, for example, to suggest new products to be purchased.

3) Test Results

The elements of application that constitute the tested prototype were deployed on separate computers. Each system module (two stores and one warehouse) is composed of "CEP Mechanism + RFID Middleware + EPCIS", as illustrated in figure 1.

In the first test scenario (warehouse application), we identified an average of 390 events processed per second, including basic events (EPCIS) and complex events. In the event detection capability (Figure 7), the results show that for the first test scenario, we detected an average of 1.03 basic events for each RFID event generated. For complex event detection capability, the average ratio is 0.06 complex events detected for each RFID event generated. In other words, less than one complex event was detected for each 10 RFID events generated. In the best-case situation, using a load of 50,000 events generated, it is possible to detect one complex event for every 10 RFID events generated.

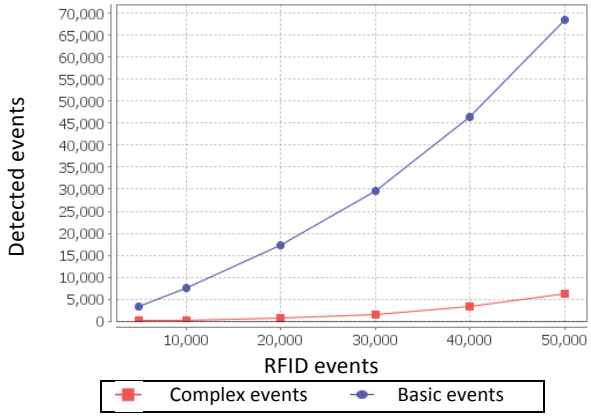

Figure 7: Scenario 1 – Events detected.

We observed that the first test scenario is working practically in the contextualization of RFID events into EPCIS events (1.03/1 basic event), and not much in the detection of complex events.

Figures 8 and 9 correspond to tests for the second test scenario (3 applications: one warehouse and two stores).

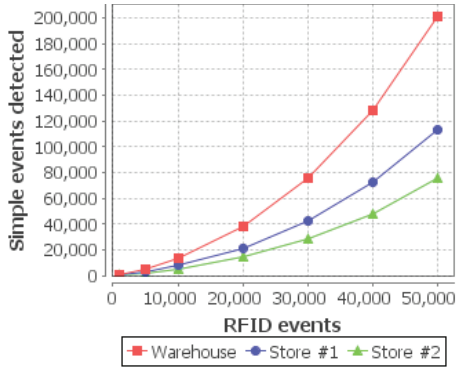

Figure 8: Scenario 2 – Basic events detected.

The average processing time of events of the three applications is 57 events processed per second. Regarding the processing time of the first test scenario, the applications of the second scenario had a worse performance, which is justified by the increased complexity of the processing rules involved, as described in the test methodology.

Regarding to the average capability of detecting basic event (EPCIS events) (Figure 8), the tests indicate a detection rate of 2.6 basic events detected for each RFID event generated. To the average detection capability of complex events (Figure 9), the tests indicate 0.3 complex events detected for each RFID event generated.

We noticed that the test applications of the second scenario are contextualizing more RFID events than the first scenario. Moreover, in the second scenario the applications are detecting one complex event for each 3 RFID events generated, resulting in a performance gain of 70% in the ability to identify complex events when compared to the first scenario.

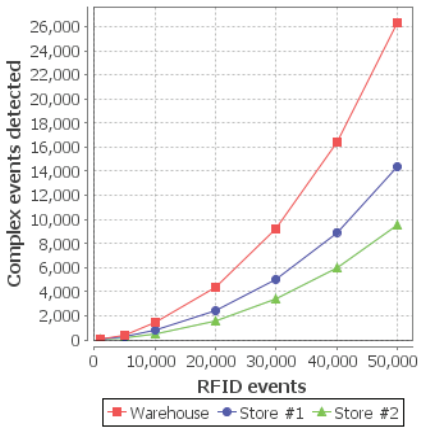

Figure 9: Scenario 2 – Complex events detected.

V. DISCUSSION

A. Vulnerability Analysis

Regarding to the vulnerability in the performance of the CEP mechanism, the tests indicate that increasing the number of test data (RFID events generated) and business rules applied to the CEP mechanism, the ability to detect events (basic and complex events) also increases. However, there is also an increase in the processing time.

This elevation in the processing time is not only due to the increase in the number of rules used, but also in the delay caused by the communication network in the event sharing between applications (event notification service). However, the communication overhead does not impact in the ability of event detection.

The CEP engine has a good performance if we consider that the generated test load was much higher than normal (50 or 60 complex events detected per second). Moreover, the tests were applied in conventional computer machines. Therefore, better performance rates can be achieved when applied in high-performance computer machines.

B. Architectural Features Analysis

Regarding to the evaluation of artifacts and software mechanisms that comprise our framework, Table 1 presents a brief comparison between the framework and the architecture of Fosstrak. For this analyze, we used nine issues related to design activities, development and integration of RFID applications.

In our framework, the complexity of integration between CEP mechanism and implementations of ALE and EPCIS is considered "low". We use configuration files describing interfaces for integration that are loaded into the framework through the administrative interface. However, in the case of Fosstrak, this integration is considered "complex" because it requires the implementation of the desired integration via programming of the capture application.

Regarding to the support of complex event processing, both systems offer CEP mechanisms to process complex events. Fosstrak only uses the rule engine to capture and contextualize RFID events. Whereas in our work, the CEP engine is used not only to capture and contextualize RFID events, but also to capture and notify other types of events such as EPCIS events that are shared between organizations through the event capture interface and event notification service.

Table 1 – Features analyzed.

Issue	Fosstrak	Framework
1. Create an EPCIS object	37 lines	3 lines
2. Capture an EPCIS event	4 lines	Automatic
3. Travel ALE reports	28 lines	Automatic
4. Query EPCIS	20 lines	1 line
5. Put items or package in hierarchical structures	Complex	1 line
6. Integration with ALE (middleware) and EPCIS implementations	Complex	Low
7. Complex event processing support	Yes	Yes
8. Complexity to develop new capture applications	High	Mid
9. Impact of changes in business processes	Low	Low

Regarding to the complexity of developing new event capture applications, in Fosstrak this activity has a "high" complexity due to lack of tools (classes and specific methods) to assist in building new capture applications. This occurs even Fosstrak allowing integration with CEP rule engine. Our work requires a "mid" complexity because it demands a high technical knowledge to use the rule definition language of the CEP engine, since capturing and event contextualization activities are based on business rules. As both systems have CEP rule engines, the impact of changes in the business processes is considered "low" because they change rules directly in the database instead of changing the application code.

C. Advantages compared to existing works

When compared with major existing RFID systems (Section 2), our work offers some improvements while complying with the key requirements involved.

For handling large-volume of data and limited lifetime, our framework uses CEP to guide the storage activities partitioning the data in active and inactive data type. Active data are addressed in the working memory of the rule engine or in EPCIS repository. However, inactive data (those events who have ended their life cycle) are removed from the EPCIS repository and stored in another database. It improves the performance of the system avoiding unnecessary data queries.

For handling temporal and dynamic data, our framework meets these requirements through the CEP mechanism, which is based on the temporal ability of Drools.

Regarding to the scalability of the framework, our work offers both intra and inter-organizational scalability level. Through the CEP network service, the framework can scale either in terms of large-volume data provided by various middleware systems involved in the same organization (intra-organizational), as through distributed business process integrated with various CEP mechanisms and middleware systems involved (inter-organizational).

Finally, the administrative capability of the framework is also based on the organizational profile. With profiles parameterization, it is possible to automatically configure, either logical (rules for event processing, interfaces to capture and event notification, and the relationship between CEP, middleware and EPCIS) as physical aspects of the system (abstraction of the network infrastructure, i.e. logical readers).

VI. Conclusion and Future Work

This work presented an RFID software framework whose goal is to raise the inter-organizational integration capability of an RFID middleware system architecture through CEP mechanisms that cooperate with each other through event notification services.

Although the use of CEP engines in RFID applications is not considered an innovative strategy, the innovation in our work takes place using the same CEP mechanism for both "improving the processing of complex business events among distributed organizations" and "to fill the lack of a general event capture application in the business process automation".

The idea is to offer CEP mechanisms that cooperate to provide services for capturing, processing and integration of distributed RFID event data increasing the functional capability of RFID middleware systems.

Experimental results show that the proposed framework improves activities of design and development of RFID applications and adds new features for integrating, managing and sharing data from RFID events.

As future work, we can identify some issues that should be treated to improve the performance of the intra-organizational CEP mechanism and also the management and integration ability of the framework with business processes.

We believed that improvements in terms of load balancing between CEP mechanisms and information sources (RFID middleware systems) could increase the event processing capability in large-scale organizations.

Regarding to improvements in the management and integration capability of the framework, the creation of a modeling tool that assists stakeholders in setting rules and also in shaping the operational environment is also an important and innovative contribution. The creation of this tool will allow the automation of various activities that are manual today in the framework, such as: integration with ALE (middleware) and EPCIS implementations, definition and creation of business rules and changes in business processes.

VII. Acknowledgments

The Research and Projects Financing (FINEP) supports this work in the scope of the project SRAM under Grant nº 0108031000.

References

[1] Ziekow, H.; Gunther, O. **Sharing RFID and Complex Event Data Among Organizations**. In: Information Systems Frontiers Special Issue on RFID, 2009.

[2] Chawathe, S. S.; Krishnamurthy, V.; Ramachandran, S.; Sarma, S. "**Managing RFID Data**". In: Proceedings of the 30th International Conference on Very Large Data Bases, 2004, pp. 1189-1195.

[3] Wang, F.; Liu, S.; Liu, P.; Bai, Y. **Bridging Physical and Virtual Worlds: Complex Event Processing for RFID Data Streams**. In: Proceedings of the 10th International Conference on Extending Database Technology (EDBT'2006), Munich, Germany, 2006.

[4] Jeffery, S. R.; Garofalakis, M.; Franklin, M. J. "**Adaptive Cleaning for RFID Data Streams**". In: Proceedings of the 32nd International Conference on Very Large Data Bases, 2006, p: 163-174.

[5] Hu, Y.; Sundara, S.; Chorma, T.; Srinivasan, J. **Supporting RFID-Based Item Tracking Applications in Oracle DBMS Using a Bitmap Datatype**. In: Proceedings of the 31st International Conference on Very Large Data Bases (VLDB'05), Trondheim, Norway, 2005.

[6] Gonzalez, H.; Han, J,; Li, X. **FlowCube: Constructing RFID FlowCubes for Multi-dimensional Analysis of Commodity Flows**. In: Proceedings of the 32nd International Conference on Very Large Data Bases (VLDB'06), Seoul, Korea, 2006.

[7] SAP. **SAP Auto-ID Infrastructure**. http://www.sap.com/solutions/netweaver/autoidinfrastructure.epx, visited on 25/11/2010.

[8] Prabhu, B.S.; Su, X.; Ramamurthy, H.; Chu, C.; Gadh, R. **WinRFID - A Middleware for the Enablement of Radio Frequency Identification (RFID)-Based Applications**. In: Mobile, Wireless and Sensor Networks: Technology, Applications and Future Directions, Wiley–IEEE Press, New York, 2006, pp. 313–336.

[9] Sun Microsystems. **Sun Java System RFID Software**. http://java.sun.com/developer/technicalArticles/Ecommerce/rfid/sjsrfid/RFID.html, visited on 25/11/2010.

[10] Fosstrak, **An Open Source RFID Platform**, www.fosstrak.org, Last accessed 25/11/2010.

[11] Wu, E.; Diao, Y, Rizvi, S. **High-Performance Complex Event Processing Over Streams**. In: Proceedings of the 2006 ACM SIGMOD International Conference on Management of Data, 2006.

[12] Jeng, J. J,; Flaxer, D,; Kapoor, S. **RuleBAM: A Rule-based Framework for Business Activity Management**. In: Proceedings of the IEEE International Conference on Service Computing (SCC'04), Shanghai, China, 2004. pp. 262–70.

[13] McCarthy, D.; Dayal, U. **The Architecture Of an Active Database Management System**. ACM SIGMOD, Vol. 18, 1989, pp.215–24.

[14] Zang, C,; Fan, Y,; Liu, R. **Architecture, Implementation and Application of Complex Event Processing in Enterprise Information Systems Based on RFID**. Information Systems Frontiers, Vol. 10, 2008, pp. 543–553.

[15] Wu, J.; Wang, D.; Sheng, H. **ECA Rule-Based RFID Data Management**, In: First annual RFID Eurasia, 2007, pp. 1-5.

[16] Wang, Y.; Yang, S. **Plan Based Distributed Complex Event Processing for RFID Application**, In: Proceedings of the International Conference on Computational Intelligence and Software Engineering, 2009, pp. 1-4.

[17] Kim, S.; Moon, M.; Kim, S., Yu, S.; Yeom, K. **RFID Business Aware Framework for Business Process in the EPC Network**, In: Proceedings of the 5th ACIS International Conference on Software Engineering Research, Management & Applications, 2007, pp. 468-475.

[18] Welbourn, E.; Khoussainova, N.; Letchner, J, Li, Y. **Cascadia: A System for Specifying, Detecting, and Managing RFID Events**. In: Proceeding of the 6th International Conference on Mobile Systems, Applications, and Services, 2008, pp. 281-294.

[19] Bali M. **Drools JBoss Rules 5.0 Developer's Guide**. Birmingham, UK: Packt Publishing; 2009.

[20] Yao, W; Chu, C.; Li, Z. **Leveraging Complex Event Processing for Smart Hospitals Using RFID**. Journal of Network and Computer Applications, 2010, doi: 10.1016/j.jnca2010.04.020.

[21] Forgy, C. L. **Rete: A Fast Algorithm for the Many Pattern/Many Object Pattern Match Problem**. In Expert systems, IEEE Computer Society Press, Los Alamitos, CA, USA, 1991, pp. 324-341.